Babel

Objectifs

L'objectif de cette démonstration est l'utilisation de babel pour "transformer" un code es6+ pour qu'il soit utilisable par de vieux navigateurs qui ne supporte pas cette version de ECMAscript

Mise en place

La mise en place effectuée dans la démonstration sur les classes ES6 peut être utilisé.

- Créer un fichier index.html dans un dossier *demonstrations/module02*
- Créer un fichier script.js dans un dossier *javascript*
- Faire le lien entre le fichier html et le fichier javascript

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>JOO</title>
</head>
<body>
<script src="script.js"></script>
</body>
</html>
```

ES6

le fichier javascript de la démonstration sur les classes ES6 peut être utilisé.

- Ecrire du code javascript ES6+. Par exemple, en utilisant les classes :

```
class Livre {
    #titre;
    #sous_titre;
    #annee;
    #auteur;

    constructor(titre, sous_titre, annee, auteur) {
        this.#titre = titre;
        this.#sous_titre = sous_titre;
        this.#annee = annee;
```

```
        this.#auteur = auteur;
    }

    getDescription() {
        return `${this.#titre} a été écrit par ${this.#auteur} en ${this.#annee}`;
    }
}

const livre5 = new Livre("Explorer Kaamelott", "Les dessous de la table ronde", 2021,
"Clément Pelissier");
console.dir(livre5);
console.log(livre5.getDescription());
```

NPM

- Créer un projet géré par NPM avec la commande :

```
npm init -y
```

- Installer les paquets ad hoc :

```
npm install --save-dev @babel/core @babel/cli @babel/preset-env
```

Babel

core est le paquet principal de babel, cli est l'interface en ligne de commande et un preset est en ensemble de règles de "transformation" que l'on va pouvoir surcharger.

- Utiliser l'interface en ligne de commande pour tenter de convertir du code es6+

```
.\/node_modules/.bin/babel script.js --out-dir lib
```

- Faire le constat qu'un fichier javascript a bien été créé dans le dossier lib mais qu'il est strictement identique au fichier original.

En effet, aucune règle n'a été définie donc il n'a rien transformé. On peut lui demander de ne s'occuper que des classes, ou bien que des méthodes fléchées mais la plupart du temps on va utiliser des presets ou des fichiers de configuration pour se faciliter le travail.

Presets

```
.\/node_modules/.bin/babel script.js --out-dir lib --presets=@babel/env
```

- Constater la différence entre les deux fichiers

```
"use strict";

function _typeof(obj) { "@babel/helpers - typeof"; return _typeof = "function" ==
typeof Symbol && "symbol" == typeof Symbol.iterator ? function (obj) { return typeof
obj; } : function (obj) { return obj && "function" == typeof Symbol && obj.constructor
=== Symbol && obj !== Symbol.prototype ? "symbol" : typeof obj; }, _typeof(obj); }
function _classCallCheck(instance, Constructor) { if (!(instance instanceof
Constructor)) { throw new TypeError("Cannot call a class as a function"); } }
function _defineProperties(target, props) { for (var i = 0; i < props.length; i++) {
var descriptor = props[i]; descriptor.enumerable = descriptor.enumerable || false;
descriptor.configurable = true; if ("value" in descriptor) descriptor.writable = true;
Object.defineProperty(target, _toPropertyKey(descriptor.key), descriptor); } }
function _createClass(Constructor, protoProps, staticProps) { if (protoProps)
_defineProperties(Constructor.prototype, protoProps); if (staticProps)
_defineProperties(Constructor, staticProps); Object.defineProperty(Constructor,
"prototype", { writable: false }); return Constructor; }
function _toPropertyKey(arg) { var key = _toPrimitive(arg, "string"); return _typeof
(key) === "symbol" ? key : String(key); }
function _toPrimitive(input, hint) { if (_typeof(input) !== "object" || input ===
null) return input; var prim = input[Symbol.toPrimitive]; if (prim !== undefined) {
var res = prim.call(input, hint || "default"); if (_typeof(res) !== "object") return
res; throw new TypeError("@@toPrimitive must return a primitive value."); } return
(hint === "string" ? String : Number)(input); }
function _classPrivateFieldInitSpec(obj, privateMap, value) {
_checkPrivateRedeclaration(obj, privateMap); privateMap.set(obj, value); }
function _checkPrivateRedeclaration(obj, privateCollection) { if (privateCollection
.has(obj)) { throw new TypeError("Cannot initialize the same private elements twice on
an object"); } }
function _classPrivateFieldGet(receiver, privateMap) { var descriptor =
_classExtractFieldDescriptor(receiver, privateMap, "get"); return
_classApplyDescriptorGet(receiver, descriptor); }
function _classApplyDescriptorGet(receiver, descriptor) { if (descriptor.get) { return
descriptor.get.call(receiver); } return descriptor.value; }
function _classPrivateFieldSet(receiver, privateMap, value) { var descriptor =
_classExtractFieldDescriptor(receiver, privateMap, "set"); _classApplyDescriptorSet
(receiver, descriptor, value); return value; }
function _classExtractFieldDescriptor(receiver, privateMap, action) { if (!privateMap
.has(receiver)) { throw new TypeError("attempted to " + action + " private field on
non-instance"); } return privateMap.get(receiver); }
function _classApplyDescriptorSet(receiver, descriptor, value) { if (descriptor.set) {
descriptor.set.call(receiver, value); } else { if (!descriptor.writable) { throw new
TypeError("attempted to set read only private field"); } descriptor.value = value; } }
var _titre = /*#__PURE__*/new WeakMap();
var _sous_titre = /*#__PURE__*/new WeakMap();
var _annee = /*#__PURE__*/new WeakMap();
var _auteur = /*#__PURE__*/new WeakMap();
var Livre = /*#__PURE__*/function () {
  function Livre(titre, sous_titre, annee, auteur) {
```

```
    _classCallCheck(this, Livre);
    _classPrivateFieldInitSpec(this, _titre, {
      writable: true,
      value: void 0
    });
    _classPrivateFieldInitSpec(this, _sous_titre, {
      writable: true,
      value: void 0
    });
    _classPrivateFieldInitSpec(this, _annee, {
      writable: true,
      value: void 0
    });
    _classPrivateFieldInitSpec(this, _auteur, {
      writable: true,
      value: void 0
    });
    _classPrivateFieldSet(this, _titre, titre);
    _classPrivateFieldSet(this, _sous_titre, sous_titre);
    _classPrivateFieldSet(this, _annee, annee);
    _classPrivateFieldSet(this, _auteur, auteur);
  }
  _createClass(Livre, [{
    key: "getDescription",
    value: function getDescription() {
      return "".concat(_classPrivateFieldGet(this, _titre), " a \xE9t\xE9 \xE9crit par
").concat(_classPrivateFieldGet(this, _auteur), " en ").concat(_classPrivateFieldGet
(this, _annee));
    }
  }]);
  return Livre;
}();
```

Configuration

L'ensemble des subtilités du fichier de configuration de babel est disponible ici.

Pour l'heure, on va simplement lui donner les versions cibles des navigateurs web pour lesquelles on souhaite faire fonctionner notre site web.

- Créer un fichier *babel.config.json* à la racine du projet
- Indiquer les versions de navigateur que l'on souhaite supporter

```
{
  "presets": [
    [
      "@babel/preset-env",
      {
        "targets": {
```

```
          "edge": "11",
          "firefox": "60",
          "chrome": "67",
          "safari": "11.1"
        }
      }
    ]
  ]
}
```

- Relancer la commande sans le preset :

```
.\/node_modules/.bin/babel script.js --out-dir lib
```